

Ziel- und kundenorientierte Anforderungserstellung mit Abstraktionsebenen als zentraler Erfolgsfaktor bei der Entwicklung von Kfz-Software

Hannes Omasreiter, Ramin Tavakoli Kolagari

Abstract

Die Nützlichkeit einer Herangehensweise bei der Erstellung von Software-Anforderungen, die sich an den Kunden des zu entwickelnden Systems und an deren Zielen orientiert, ist in der Theorie anerkannt. In der Praxis trifft man allerdings häufig auf Systementwickler, die stark technik- und lösungsorientiert denken und daher bei der Anforderungserstellung oft direkt Lösungswege beschreiben, ohne die Frage nach den Kunden und deren Bedürfnissen zu vertiefen. Das Ergebnis dieses Verhaltens sind im schlechtesten Fall Softwarefunktionen, die den Anwenderwunsch nicht treffen. In diesem Artikel wird ein Ansatz zur Erstellung von Anforderungen vorgestellt und an einem konkreten Beispiel eines Drehzahlmessers eines Kombiinstruments erörtert, bei dem basierend auf Abstraktionsebenen eine zielgerichtete und kundenorientierte Vorgehensweise unterstützt wird. Dabei wurde ein besonderes Augenmerk darauf gelegt, dieses Konzept möglichst einfach und damit praktikabel zu halten.

Keywords

Anforderung, Ziel, Feature, Use Case, Abstraktionsebene

1 Einleitung

In modernen Automobilen basieren zahlreiche Innovationen auf der Fahrzeug-Software. Der Einsatz von Software kann Autofahrten beispielsweise sicherer, komfortabler, umweltfreundlicher oder unterhaltsamer machen und damit einen hohen Nutzen für die Fahrzeuginsassen schaffen. Heutigen Software-basierten Fahrzeugfunktionen wie Bremsassistent, ESP (Elektronisches Stabilitätsprogramm) oder Abstandsregeltempomat werden in Zukunft eine Vielzahl weiterer Assistenzsysteme und Telema-tikdienste folgen. Software wird dementsprechend mehr und mehr den Wert eines Fahrzeugs mitbestimmen und damit die Kaufentscheidung beeinflussen. Über den Verkaufserfolg eines Automobils wird also zukünftig noch deutlich stärker als heute die Fahrzeug-Software entscheiden.

Damit stellt sich für einen Automobilhersteller die Frage, wie Software gestaltet werden muss, um zum Erfolg eines Kraftfahrzeugs optimal beitragen zu können. Zur Beantwortung dieser Frage ist es notwendig, die Ziele zu kennen, die bei der Entwicklung von Fahrzeug-Software erreicht werden sollen. Folgende vier Kernziele sind dabei wesentlich:

- **Kundenorientierung:**

Dieser Punkt stellt ein übergreifendes Ziel für die übrigen Ziele dar und beinhaltet die Anforderung, einen möglichst hohen Wert für den Kunden zu schaffen. Software-Funktionen sollen also die Wünsche des Kunden möglichst gut erfüllen.

- **Innovation:**

Jede Software-Entwicklung ist innovativ. Innovation schafft aber nur dann einen Mehrwert, wenn die Neuerung auch den Nutzen für den Anwender erhöht. Die Auswahl der nützlichsten Software-Funktionen ist damit äußerst wichtig.

- **Qualität:**

Neben der innovativen Nützlichkeit einer Software-Funktion ist es für den Anwender ebenso wichtig, dass die Funktion bestimmte Qualitätskriterien erfüllt, also beispielsweise einfach zu bedienen oder in jeder Situation sicher ausführbar ist.

- **Kosten und Termine:**

Die Kosten von Software sollen möglichst niedrig sein und die Entwicklung soll innerhalb einer möglichst kurzen Zeitspanne erfolgen.

Die Nützlichkeit einer Herangehensweise bei der Erstellung von Software, die sich an den Kunden des zu entwickelnden Systems und an deren Zielen orientiert, ist in der Theorie und teilweise auch in der Praxis anerkannt [1, 2]. Dennoch sind in der Praxis häufig außer Kosten- und Terminvorgaben kaum weitere explizite Ziele vorzufinden, und selbst die wenigen Vorgaben können oft nicht wie geplant erfüllt werden. Die Frage nach den Kunden und dessen Zielen rückt vielfach zugunsten einer technik- und lösungsorientierten Sichtweise in den Hintergrund. Gerade wegen des hohen Zeitdrucks erscheint es Systementwicklern oft sinnvoller, schnelle Lösungen zu produzieren und sich nicht mit Fragen aufzuhalten, die solche Lösungen aus deren Sicht nur verzögern. Das Fehlen eines zunächst als zeitraubend und wenig nützlich erachteten Nachdenkens über die Kundenziele führt aber leicht zu Ergebnissen, die letztlich dem Kunden keinen wirklichen Mehrwert bringen, weil sie an seinen Bedürfnissen vorbeigehen.

Um diese Situation zu verbessern, benötigt ein Entwickler in erster Linie Hilfestellung für eine verbesserte Herangehensweise bei der Anforderungserstellung, weil in diesem Prozessschritt die wesentlichen Weichen für den Projekterfolg gestellt werden. In dieser Arbeit wird ein solches Verfahren vorgestellt, das im Rahmen des ITEA Empress¹ Projektes entwickelt wurde. Der Ansatz basiert auf dem Grundgedanken, eine ziel- und kundenorientierte Erstellung von Software-Anforderungen durch geeignete Abstraktionsebenen zu unterstützen.

Die bekannten Vorteile zielorientierter Methoden bei der Anforderungserstellung wie etwa die Unterstützung beim Auffinden der tatsächlich notwendigen und der Vermeidung unnötiger Anforderungen (vgl. etwa [1]), werden bei diesem Konzept verknüpft mit den Vorteilen, die eine an den Systemkunden orientierte Klassifizierung und Abstraktion von Anforderungen mit sich bringt. Dabei wurde insbesondere darauf geachtet, das Konzept möglichst einfach und verständlich zu halten, damit es möglichst gute Chancen hat, in der Praxis akzeptiert und tatsächlich dauerhaft eingesetzt zu werden. Im folgenden Abschnitt werden die grundsätzlichen Ideen dieses Konzeptes vorgestellt. Anschließend wird eine beispielhafte Umsetzung anhand von Anforderungen für einen Drehzahlmesser in einem Kombigerät eines Pkw erläutert.

¹ <http://www.empress-itea.org/index.html>

2 Konzept der ziel- und kundenorientierten Anforderungserstellung mit Abstraktionsebenen

Das von uns vorgeschlagene Rahmenwerk für eine Anforderungserstellung, die an Kunden und deren Zielen ausgerichtet ist, basiert auf drei Abstraktionsebenen (auch: -schichten, oder -stufen), wobei jede Ebene eine grundsätzlich andere Sichtweise auf ein Anforderungsdokument bereitstellt, nämlich die Sicht des Managements, des Anwenders und des Systementwicklers. Obwohl es in der Regel mehr als diese drei Interessensgruppen zu berücksichtigen gilt, halten wir die drei genannten Sichten für ausreichend. Jede Anforderung ist einer dieser Anforderungsebenen zugeordnet und gehört damit zu einer der drei folgenden Klassen (ähnliche Bezeichnungen sind etwa in [3] zu finden):

Business-Anforderungen sind die Anforderungen, für die sich hauptsächlich das Management interessiert. Entsprechend müssen die Anforderungen auf dieser Ebene so beschrieben sein, dass sie in möglichst kurzer Zeit möglichst leicht zu verstehen sind und möglichst ohne dass spezielles technisches Verständnis dazu notwendig ist.

User-Anforderungen sind die Anforderungen, für die sich hauptsächlich der spätere Anwender des Systems interessiert. Die Darstellung dieser Anforderungen soll so sein, dass sie das System aus der Sicht eines Anwenders beschreibt, also in einer für den technischen Laien verständlichen Sprache und ohne Implementierungsdetails, die den Anwender nicht interessieren.

System-Anforderungen sind die Anforderungen, für die sich hauptsächlich der Systementwickler interessiert. Diese Ebene muss alle Anforderungen an das zu entwickelnde System (also etwa auch notwendige Realisierungsvorgaben) vollständig abdecken und ist damit in der Regel nur für technische Experten vollständig verständlich.

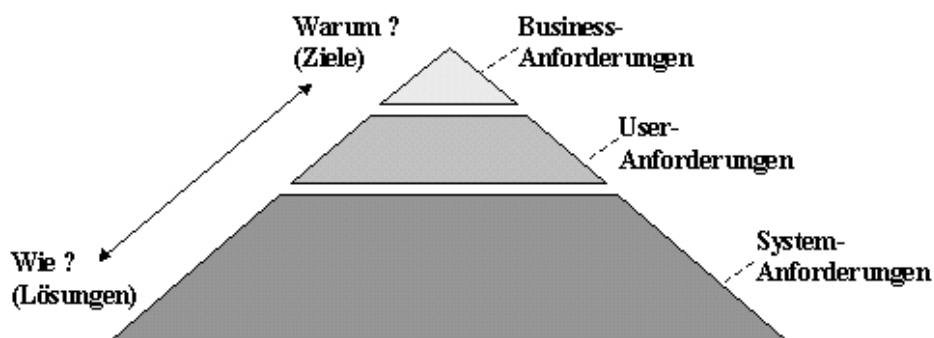


Abbildung 1: Abstraktionsschichten-Pyramide

Grundsätzlich ist bei dieser Anforderungsstruktur, die in der Abbildung 1 als Abstraktionsschichten-Pyramide dargestellt ist, eine Anforderungsschicht umso lösungsorientierter, je tiefer die Schicht in der Pyramide platziert ist, und umso zielorientierter, je höher die Schicht platziert ist. Jede Anforderung einer Schicht (außer der System-Anforderungsschicht) muss in einer untergeordneten Schicht berücksichtigt sein. Nach Möglichkeit sollte auch für jede Anforderung einer Schicht (außer der Business-Anforderungsschicht) mindestens eine Anforderung in einer übergeordneten Schicht vorhanden sein.

Das gezeigte Grundkonzept der Abstraktionsschichten-Pyramide sieht also vor, neben den eigentlichen System-Anforderungen zusätzlich Anforderungen aus Sicht des Kunden und des Projekt-Managements zu erheben und jeweils in einer eigenen Informationseinheit abzulegen. Idealerweise sollen zuerst die Business-Anforderungen komplett aufgestellt werden, danach die User-Anforderungen und zum Schluss die System-Anforderungen. In bestimmten Situationen kann allerdings auch eine andere Reihenfolge Sinn machen, beispielsweise dann, wenn bereits umfangreiche System-Anforderungen vorliegen. Mit welchen Mitteln die Anforderungen auf den drei Schichten beschrieben werden, kann je nach Projektumfeld unterschiedlich sein. Eine von uns vorgeschlagene Möglichkeit, die Ebenen zu instanziierten, verdeutlicht die Abbildung 2.

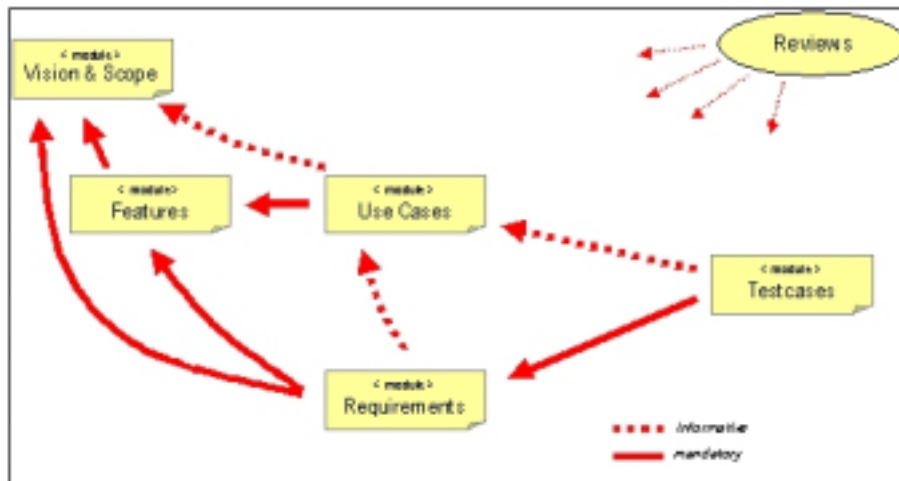


Abbildung 2: Mögliches Informationsmodell zur Abstraktionsschichten-Pyramide

Das dargestellte Informationsmodell definiert Informationseinheiten und Beziehungen zwischen diesen Einheiten. Business-Anforderungen werden durch ein Vision-and-Scope-Modul abgebildet, das die Vision und den Umfang des Systems aus Management-Sicht aufnimmt. Features² dienen als zentrale Beschreibungsmethode für User-Anforderungen und Use Cases helfen optional dabei, die User-Anforderungen zusätzlich verständlich zu machen. Jedes Feature ist mit mindestens einem übergeordneten Ziel verbunden, jeder Use Case mit mindestens einem Feature und optional auch mit einem übergeordneten Ziel. Die System-Anforderungen werden in einem Requirements-Modul abgelegt und zu jeder Anforderung in diesem Modul existiert mindestens ein übergeordnetes Feature oder Ziel des Vision-and-Scope-Moduls und optional auch ein Use Case.

Durch die starke Fokussierung der User-Anforderungen auf Features kann bei dieser Instanziierung auch von einem Feature-orientierten Ansatz gesprochen werden [4]. Features eignen sich gut zur knappen und statischen Darstellung der wichtigen Kundenfunktionen eines Systems. Wenn auch Interesse an den dynamischen Abläufen von Funktionen besteht (z.B. besonders bei neuartigen Funktionalitäten), ist der ergänzende Einsatz von Use Cases empfehlenswert [5, 6].

Über die durchgehende Verlinkung der Anforderungsmodule ist automatisch eine Nachvollziehbarkeit und Begründbarkeit für jede Anforderung, insbesondere auf unterer Ebene, gegeben. Das Beispiel deutet zusätzlich an, wie durch Reviews aller Informationsmodule und geeignete Testfälle die aufgestellten Anforderungen und die implementierte Software überprüft werden können. Eine Ableitung von Testfällen aus Use Cases bietet die Möglichkeit, Tests aus der Sicht eines Kunden durchzuführen, so dass beim Systemtest im Idealfall keine Szenarien unberücksichtigt bleiben, die für den Anwender wichtig sind.

Die Vorteile eines Ansatzes zur Anforderungserstellung, der auf einer solchen Informationsstruktur basiert, gegenüber einem Verfahren, das sich hauptsächlich auf die System-Anforderungen konzentriert, liegen in mehreren Punkten:

- Die Verständlichkeit eines Anforderungsdokuments steigt durch die Business- und User-Anforderungen deutlich an und unterstützt damit die Unterscheidung nach Interessensgruppen und die Trennung von Zielen und Lösungen.
- Die Ableitung von Anforderungen aus übergeordneten Zielen macht jede Systemanforderung klar begründbar und fördert die Konzentration darauf, die richtigen Ziele und damit auch die richtigen Systemfunktionen zu finden.
- Die zielorientierte Vorgehensweise kann dazu beitragen, dass durch klare Einigungen auf Ziele unnötige Diskussionen über Lösungen, die den vereinbarten Zielen nicht gerecht werden, vermieden werden können.

² Features beschreiben essentielle Eigenschaften des Systems. Es kann sich dabei sowohl um funktionale als auch nicht-funktionale Anforderungen handeln.

- Die Verständlichkeit und Verlinkung von Anforderungen ist hilfreich für deren Wiederverwendung in anderen Anforderungsdokumenten.

3 Beispiel einer ziel- und kundenorientierten Anforderungserstellung mit Abstraktionsebenen

Im folgenden werden wir das oben beschriebene Konzept an einem Beispiel aus der Pkw-Elektrik/Elektronik-Entwicklung bei DaimlerChrysler verdeutlichen. Es handelt sich hierbei um eine fiktive Spezifikation des Drehzahlmessers, ein Instrument, das eingebettet im Kombiinstrument eines Fahrzeugs die Anzahl der Umdrehungen des Motors pro Minute anzeigt (Englisch: rotations per minute, RPM), indem ein Zeiger auf Werte einer Kreisskala zeigt.



Abbildung 3: Display des Drehzahlmessers

Wir haben den Drehzahlmesser als Beispiel gewählt, weil die Anforderungen und das Verhalten dieses Instruments aus Sicht des Fahrers offensichtlich und einleuchtend sind. Außerdem ist es allgemein bekannt, was man von einem Drehzahlmesser zu erwarten hat. Allerdings ist der Drehzahlmesser komplex genug, um schon in diesem Rahmen die Vorteile unseres Konzepts zu verdeutlichen. Zu guter Letzt ist der Drehzahlmesser ein perfektes Beispiel, um die Komplexität von Komponenten in modernen Fahrzeugen zu illustrieren, die auf den ersten Blick sehr einfach erscheinen. Die hier angegebenen Anforderungen reichen bei weitem nicht aus um einen Drehzahlmesser zu spezifizieren. Sie dienen hier nur zur Verdeutlichung, wie die Anforderungen auf den verschiedenen Abstraktionsebenen aussehen könnten.

3.1 Business-Anforderungen - Vision & Scope des Drehzahlmessers

Da diese Abstraktionsebene vor allem der Kommunikation zwischen Entwickler und Management dient, sind die Anforderungen leicht verständlich verfasst und sehr knapp gehalten. Zunächst geben wir die *Ziele* an, die aus Sicht des Managements von Bedeutung sind:

ID	Ziel (Vision)	Beschreibung
Z-1	Komfortabler Blick auf die Anzeige der Umdrehungen des Motors pro Minute	Der Fahrer hat einen optimalen Blick auf den Drehzahlmesser, der regelmäßig aktuelle Informationen anzeigt.
Z-2	Maximiere die Robustheit und Lebensspanne des Drehzahlmessers	Die durchschnittliche Lebensdauer des Drehzahlmessers soll die durchschnittliche Lebensdauer des gesamten Fahrzeugs überschreiten; eine Nichterfüllung dieser

		Anforderung ist nur in weniger als in einem von 10.000 ausgelieferten Fahrzeugen tolerierbar.
Z-3	Wertanmutende Erscheinung	Das Display und das Verhalten des Drehzahlmessers sollen sportlich, agil und wertanmutend sein.
Z-4	Minimiere Energieverbrauch	den Der Drehzahlmesser soll nur die geringst vertretbare Menge an Energie verbrauchen.
Z-5	Sicherheit	Der Drehzahlmesser soll eine angemessene Unterstützung für die Sicherheit bieten.

Tabelle 1: Ziele des Drehzahlmessers – Beispiel für Business-Anforderungen

Neben den Zielen gibt es außerdem *Einschränkungen* der Menge möglicher Lösungen und des Entwicklungsprozesses, den Scope:

ID	Einschränkung (Scope)	Beschreibung
E-1	Entwicklungsprozess	Die Entwicklung des Drehzahlmessers soll mindestens auf SPICE Level 3 erfolgen (ISO 15504).
E-2	Verantwortlichkeit	Die Verantwortlichkeit für Elektrik/Elektronik der Baureihe EMP 01 liegt bei Herrn Düsentrieb. Endgültige Entscheidungen werden nur mit seiner Zustimmung gefällt.

Tabelle 2: Einschränkungen des Drehzahlmessers – Beispiel für Business-Anforderungen

3.2 User-Anforderungen - Features & Use Cases des Drehzahlmessers

Im folgenden wollen wir den Drehzahlmesser als Black Box betrachten, um die dem Benutzer sichtbaren Funktionalitäten und das Verhalten zu beschreiben. Zunächst listen wir *Features* (Charakteristika) des Drehzahlmessers auf, danach einen zugehörigen *Use Case* (Anwendungsfall). Die Features sind mit den Zielen des Drehzahlmessers auf der Ebene der Business-Anforderungen verlinkt.

ID	Feature	Beschreibung	Ziele
F-1	Display der Umdrehungen pro Minute des Drehzahlmessers	Das Display des Drehzahlmessers besteht aus einer Kreisskala und einem Zeiger, der die Umdrehungen des Motors pro Minute anzeigt.	Z-1
F-2	Dämpfung	Die Anzeige der Motorumdrehungen soll gedämpft erfolgen.	Z-1, Z-3
F-3	Warnbereich	Es befindet sich auf der Skala ein rot hinterlegter Warnbereich, der zu hohe Motordrehzahlen anzeigt.	Z-5

Tabelle 3: Features des Drehzahlmessers – Beispiel für User-Anforderungen

ID	UC-1
Use Case (Anwendungsfall):	Zeige die Umdrehungen des Motors pro Minute
Verwendete Features:	F-1, F-2
Handelnder:	Fahrer
Beschreibung:	Der Fahrer startet das Fahrzeug und fährt, während er auf die Anzeige der Motorumdrehungen pro Minute achtet.
Spezifikation der Ablaufschritte:	
<p>Der Fahrer steigt in das Fahrzeug ein.</p> <p>Der Fahrer startet das Fahrzeug.</p> <p>Das Fahrzeug ist angeschaltet und der Zeiger der Anzeige des Drehzahlmessers bewegt sich gedämpft von der technischen Anfangsposition an die Anfangsposition der Kreisskala (0 min^{-1}).</p> <p>Der Drehzahlmesser stellt kontinuierlich die Umdrehungsgeschwindigkeit des Motors fest und zeigt sie durch den Zeiger auf der Kreisskala an.</p> <p>Der Fahrer schaltet das Fahrzeug aus.</p> <p>Das Fahrzeug ist ausgeschaltet und der Zeiger fällt gedämpft zurück zur technischen Anfangsposition.</p>	

Tabelle 4: Anwendungsfall des Drehzahlmessers ohne Ausnahmespezifikation – Beispiel für User-Anforderungen

3.3 System-Anforderungen - Details des Drehzahlmessers

Die System-Anforderungen spezifizieren den Drehzahlmesser auf sehr detaillierte Weise, was bedeutet, dass alle Business- und User-Anforderungen durch diese erfüllt und beachtet werden müssen. Diese Abstraktionsebene der Anforderungen entspricht beispielsweise aktuellen Lastenheften der Automobilindustrie.

Hier werden wir uns auf eine beispielhafte Darstellung einer System-Anforderung beschränken, um die Lesbarkeit nicht unnötig zu erschweren. Eine typische (funktionale) System-Anforderung sieht wie im folgenden dargestellt aus:

ID	S-171
Name	Dämpfung
Eingang	<p><i>Signal:</i> Ist_Drehzahl</p> <p><i>Parameter:</i> Dämpfungskonstante_PT1</p>
Ausgang	<p><i>Signal:</i> Anzeige_Drehzahl</p>
Vorbedingung	true
Inhalt	Das Eingangssignal (Ist_Drehzahl) wird gemäß dem Wert des Parameters Dämpfungskonstante_PT1 gedämpft. Dadurch erhält man das Ausgangssignal Anzeige_Drehzahl. Dieser Wert wird zur Anzeige gebracht.

Features	F-2
Use Cases	UC-1
Erklärung	Das PT^1 Glied ist ein Verzögerungsglied erster Ordnung. Das PT^1 -Glied lässt sich aus elementaren Übertragungsgliedern (P-, I- und S-Gliedern) aufbauen. Die Sprungantwort des PT^1 Gliedes lässt sich sehr einfach mit der Laplacetransformation berechnen. Die Sprungantwort hat einen abgeflachten Kurvenverlauf.
Kommentare	Hier ist eine Änderung angedacht: Die Eingangssignale könnten auch durch zwei PT^1 Glieder gedämpft werden. Vorteil: Bessere Dämpfung. Nachteil: Es müssen die Werte zweier Dämpfungskonstanten verwaltet werden.
Autor	Daniel Düsentrieb
Status	Bereit für Review

Tabelle 5: Beispiel für eine System-Anforderung

An diesem Beispiel haben wir exemplarisch dargestellt, wie sich unser Konzept umsetzen lässt. Die System-Anforderung Dämpfung lässt sich aus den Zielen „Komfortabler Blick“ und „Wertanmutende Erscheinung“ ableiten. Die detaillierte System-Anforderung beschreibt dann das Feature Dämpfung aus der Sicht des Systementwicklers.

Vielfach wird in der heute üblichen Praxis im wesentlichen nur die Ebene der System-Anforderungen beschrieben; demjenigen, der die Anforderungen liest (z.B. um sie zu implementieren), ist dadurch häufig nicht transparent, warum die Anforderungen gerade diesen und nicht einen anderen Inhalt haben. Außerdem kann er nicht beurteilen, ob die Anforderungen das, was sie eigentlich ausdrücken sollen, auch sinnvoll beschreiben, da er oftmals nicht den Überblick über das System hat. Diese Defizite werden durch die hier vorgestellte Herangehensweise umgangen, da sich dem Leser von derartig strukturierten Spezifikationen der Gesamtzusammenhang leichter erschließt und damit auch die Begründung und die Verständlichkeit der Anforderungen gewährleistet ist.

4 Fazit

Das hier vorgestellte Konzept für eine zielgerichtete Anforderungserstellung kann einen Entwickler dabei unterstützen, die richtigen Anforderungen aufzuschreiben und damit zur Entwicklung von Software beitragen, die dem Kunden einen hohen Nutzen bietet. Durch die Einfachheit ist die Methode schnell und ohne großen Aufwand in der Praxis umsetzbar. Erste Erfahrungen mit der Anwendung des Konzepts in konkreten Projekten bei DaimlerChrysler sind sehr positiv. Die systematische Ableitung von System-Anforderungen aus übergeordneten Zielen und die Strukturierung des Anforderungsdokuments in Abstraktionsebenen wurden in der Regel als sehr hilfreich bewertet, um Anforderungen zu erstellen und deren Sinnhaftigkeit zu überblicken.

Für eine erfolgreiche Anwendung des vorgestellten Verfahrens ist es allerdings wichtig, dass die betroffenen Entwickler die wesentlichen Ideen der Vorgehensweise verstanden haben und deren Anwendung befürworten. Andernfalls besteht die Gefahr, dass etwa die Spezifikation von Business-Anforderungen nur als zusätzlicher und unnötiger Aufwand verstanden wird, den der Entwickler möglichst gering halten möchte. Unvollständige, unklare oder falsche Anforderungen sind dann wahrscheinlich die Folge. Das Erstellen der Business- und User-Anforderungen muss jedoch sehr sorgfältig erfolgen, weil diese als Grundlage für die Ableitung der System-Anforderungen dienen.

Das beschriebene Verfahren für eine zielgerichtete Anforderungserstellung gibt keine Hilfestellung etwa beim Auffinden von Business-Zielen oder User-Anforderungen. Hier könnte in einer Weiterentwicklung beispielsweise versucht werden, das Sammeln von Anforderungen insbesondere

auf hoher Abstraktionsebene durch geeignete Mittel (wie z.B. Standardvorgaben) zu erleichtern. Da in den bisherigen Betrachtungen auf der Ebene der User-Anforderungen die funktionalen Aspekte im Vordergrund stehen, könnte ein weiteres Verbesserungspotential auch darin liegen, nichtfunktionale User-Anforderungen (wie etwa Bedienbarkeit) stärker zu berücksichtigen. Solche Verbesserungen ließen sich ohne grundsätzliche Änderungen in das vorgestellte Gesamtkonzept integrieren.

5 Literatur

- [1] Lamsweerde, A. v.: „Goal-Oriented Requirements Engineering: A Guided Tour“. Proceedings RE '01, 5th IEEE International Symposium on Requirements Engineering, Toronto, August 2001, 249-263
- [2] Rupp, C.: „Requirements-Engineering und Management“, Carl Hanser Verlag, 2002
- [3] Wiegers, K.E.: “Software Requirements”, Redmont, Microsoft Press, 1999
- [4] Chastek, G., Donohoe, P., Kang, K.C., Thiel, S.: “Product Line Analysis: A Practical Introduction”, Technical Report, CMU/SEI-2001-TR-001, June 2001
- [5] Alexander, I., Zink, T.: “An Introduction to Systems Engineering with Use Cases”, paper submitted to CCEJ, 2002
- [6] Cockburn, A.: Writing Effective Use Cases. Addison Wesley, 2001.

6 Kurzbiographie der Autoren

Hannes Omasreiter

Hannes Omasreiter studierte von 1993 - 1998 Informatik an der Technischen Universität München. Danach war er drei Jahre als Doktorand in der Pkw-Telematik-Entwicklung der DaimlerChrysler AG in Sindelfingen tätig. Seit Anfang 2001 arbeitet er als Mitarbeiter des DaimlerChrysler Forschungszentrums Ulm in der Abteilung Software-Prozessgestaltung schwerpunktmäßig im Gebiet Requirements Engineering.

Ramin Tavakoli Kolagari

Ramin Tavakoli Kolagari studierte von 1999 - 2002 Informatik an der Technischen Universität Berlin. Seit Anfang 2003 arbeitet er als Doktorand des DaimlerChrysler Forschungszentrums Ulm in der Abteilung Software-Prozessgestaltung vor allem in dem Bereich des Requirements Engineering eingebetteter Systeme und Produktlinien.

